Progressive Refinement of Raster Images

Kenneth R. Sloan, Jr.
Computer Science Department
University of Rochester

Steven L. Tanimoto
Computer Science Department
University of Washington

TR 39

November 1978

ABSTRACT

The transmission of high resolution raster images over low-bandwidth communication lines requires a great amount of time. User interaction in such a transmission environment can be frustrating. The problem can be eased somewhat by transmitting a series of low resolution approximations, which converge to the final fimage. Several methods of computing such a series of images are presented. Each is related to a particular type of pyramid data structure. They rely on the ability of the local display device to overpaint an existing image, and generally require some transmission and computation overhead. However, one of the methods requires no transmission overhead and only a small amount of local computation. A notation is introduced that permits concise descriptions of the image refinement processes.

The research described in this report was supported partially by DARPA Grant #N00014-78-C-0164.

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: TR39
2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): Progressive Refinement of Raster Images
5. TYPE OF REPORT & PERIOD COVERED: Technical report
6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): K.R. Sloan, Jr. and S.L. Tanimoto
8. CONTRACT OR GRANT NUMBER(s): N00014-78-C-0164
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Computer Science Department, University of Rochester, Rochester, N. Y. 14627
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Defense Advanced Research Projects Agency, 1400 Wilson Boulevard, Arlington, Virginia 22209
12. REPORT DATE: November 1978
13. NUMBER OF PAGES: 31 pages
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Office of Naval Research, Information Systems, Arlington, Virginia 22217
15. SECURITY CLASS. (of this report): unclassified
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report)
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)
18. SUPPLEMENTARY NOTES
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

raster display	pyramid data structure	man-machine interaction
remote graphics	approximation of images	graphics languages
image transmission	picture processing	
computer graphics	image encoding	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

ABSTRACT IS LISTED ON TITLE PAGE.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

INTRODUCTION

Raster graphics display devices are capable of reproducing very complex images. Unfortunately, they are often connected to the source of those images, a large mainframe computer, by low-bandwidth data links. This makes it difficult to interact effectively with the display when it is being used to display the images for which it was made (often full-color, typically 512*512 picture elements (pixels)). Transmitting such an image over a 1200 baud line can take half an hour, or longer. If it is being displayed on a line-by-line basis, then it may be 15 or 20 minutes before the user has any notion of what the final picture will be like.

This problem can be alleviated somewhat by sending, and displaying, a series of images which converge to the final, full resolution picture. Successive images are refinements of earlier images, and approximations to the original image. The primary advantage of such a scheme is that global structure in the image becomes apparent very early in the display process, allowing the user to begin to examine the picture, and even interrupt the display when satisfied with the approximation. The disadvantages lie in (possibly) increased storage or computation costs.

In this paper, we present several methods of computing such a series of converging images. All of these methods are based upon pyramid data structures [Tanimoto and Pavlidis, 1975]. The differences between the methods are related to the choices made in the design of pyramid data structures.

PYRAMID DATA STRUCTURES

A pyramid data structure consists of several levels, numbered 0-L, where each level is a 2-dimensional raster image. Level L is the most detailed (finest resolution) image; the others are derived from it, and are approximations to it. (Fig. 1) The value of a pixel in level k is a function of the values of the pixels in an MxN window in level k+1. Thus, the relevant parameters of a pyramid data structure are:

a) X,Y : the dimensions of Level L,

b) M,N : the dimensions of the reduction window,

c) R : the reduction rule.

Usually, the reduction window and the original image are square (M=N, X=Y=(M**L)), but these conventions can be relaxed, at some cost in computational complexity. The reduction rule can be any reasonable function of the pixels in the window (e.g., Min, Max, Mean, Median, Mode, Sum, Selection, or their extensions for handling colored pixels).

In subsequent sections we shall introduce formulas which refer to pixels in pyramids, and in order to simplify these references we denote by (k,i,j) the pixel in level k at the ith row, jth column. The set of all pyramid pixels is $P=\{(k,i,j) \mid 0 \le k \le L,\ 0 \le i < M^{**}k,\ 0 \le j < N^{**}k\}$.

NOTATION FOR RASTER OPERATIONS

We now present notation that can conveniently be used to represent many of the processes discussed in this paper. In particular, the notation will permit us to concisely describe the progressive sequences of images that our methods produce. We begin by introducing several "iconic operators." These manipulate image data by acting on grey-valued (or colored), arbitrarily-shaped regions of the picture. More precisely, each iconic operator is either a binary operator (taking two operands) or a unary operator (taking one operand), and takes operands which are "colored subsets of RR." RR (the "raster region") is an X by Y array. Thus,

$$\begin{aligned} RR = \{ & (0,0),(0,1),\ldots,(0,Y-1), \\ & (1,0),(1,1),\ldots,(1,Y-1), \\ & \vdots \\ & (X-1,0),(X-1,1),\ldots,(X-1,Y-1)\} \end{aligned}$$

Let

$$C = \{C_0, C_1, \ldots, C_{n_c}-1\}$$

be a set of colors (e.g., combinations of red, green, and blue, realizable on a particular raster display device). We assume that two special colors, black and white, are in C. Then if $S \subseteq RR$ and $f:S\to C$, we say (S,f) is a colored subset of RR. An alternative name for the concept of a colored subset of RR is "partial picture" since a colored subset of RR is in actuality completely described by the partial function f from RR to C. However, the explicit mention of the domain S of f simplifies our subsequent discussion of iconic operators.

For completeness of our basic terminology, we let F be the set of all coloring functions f and let F_S be the restriction of F to {S}. Thus the colored subsets of RR are the elements (S,f) of $2^{RR}XF$ such that dom(f) = S The <u>repainting</u> <u>operator</u>, $\underline{\%}$, is a binary iconic operator whose action is described as follows:

$$(S_1,f_1) \; \underline{\%} \; (S_2,f_2) = (S_3,f_3)$$

where

$$S_3 = S_1 \cup S_2$$

$$f_3 : S_3 \rightarrow C \quad \text{such that}$$

$$f_3(x,y) = \begin{cases} f_2(x,y) & \text{if} \quad (x,y) \in S_2 \\ f_1(x,y) & \text{if} \quad (x,y) \notin S_2 \\ & \quad \text{but } (x,y) \in S_1 \\ \text{(undefined otherwise)} & \end{cases}$$

To "repaint" using two colored subsets of RR, we make the resulting subset be the union of the two given, and define a coloring function for it as follows: the pixels common to the two original subsets get the color from the second subset. All other pixels of the new subset get the color originally assigned to them. It is easy to see that $\underline{\%}$ is associative but not commutative. Thus we write

$$\underline{\overset{n}{\underset{i=1}{\%}}}\ (S_i,f_i) = (S_1,f_1)\ \underline{\%}\ (S_2,f_2)\ \underline{\%} \ldots \underline{\%}(S_n,f_n)$$

with the understanding that the order of the terms is fixed. Another binary iconic operator makes use of distinguished colors, black and white, and an assumed color complement permutation π_C :C->C satisfying

$$\pi_c(\pi_c(c)) = c$$

and

$$\pi_c\ (\text{black}) = \text{white.}$$

The left complement operator $\underline{\angle C}$ is defined as follows:

$$(S_1,f_1)\ ¢\ (S_2,f_2) = (S_3,f_3)$$

where

$$S_3 = S_1 \cup S_2$$

$$f_3(x,y) = \begin{cases} f_1(x,y) \text{ if } (x,y) \in S_1 \text{ but } (x,y) \notin S_2 \\ f_2(x,y) \text{ if } (x,y) \notin S_1 \text{ but } (x,y) \in S_2 \\ f_1(x,y) \text{ if } (x,y) \notin S_1 \text{ and } (x,y) \in S_2 \\ \qquad \text{and } f_2(x,y) = \text{black} \\ \pi_c(f_1(x,y)) \text{ if } (x,y) \in S_1 \text{ and } (x,y) \in S_2 \\ \qquad \text{and } f_2(x,y) = \text{white} \\ \quad (\text{undefined otherwise}) \end{cases}$$

In the left complement operation a pixel in S_3 coming from S_1 keeps its original color if either the pixel does not belong to S_2 or the pixel belongs to S_2 and is colored black. If the pixel from S_1 belongs to S_2 and is colored white (by f_2) the pixel is given the complementary color in (S_3, f_3). A point in S_2 but not in S_1 keeps its color in S_3. The operator is defined so as to permit (S_2, f_2) to act as a "switch picture" to selectively complement colors of pixels from (S_1, f_1).

When all its results are defined, the left complement operator like the repainting operator is not commutative but is associative. Thus we write

$$\mathop{¢}_{i=1}^{n} (S_i, f_i)$$

with unambiguous interpretation.

We will use another operator, the "blowup" operator as an interface between pyramid and raster representations. This operator is not strictly iconic since only its result (rather than both its operand and result) is a colored subset of RR. The blowup operator is defined as follows:

$$B: PXC \rightarrow 2^{RR}XF$$

B((k,i,j),c) = (S,f) where

$s = \{ (x,y) \mid i*M**k \leq x < (i+1)*M**k,$ and $j*N**k \leq y < (j+1)*N**k \}$

and $f(x,y) = c$ (uniformly).

The blowup operator translates a pyramid pixel and its color into a corresponding region in the detailed raster region, colored with the same color.

Both the repainting operator and the left complement operator take two arguments and are thus binary. Although we shall not need it here, one may define a unary complement operator:

$$\mathcal{C}(S_1,f_1) = (S_2,f_2)$$

where

$$S_2 = S_1$$
$$f_2(x,y) = \pi_c(f_1(x,y))$$

NAIVE METHOD

Assuming that a pyramid data structure has been built, there is a straight-forward display technique which depends only on the ability of the local processor to paint rectangular regions on the screen (or in a frame buffer). The pyramid is simply transmitted "top-down". Each level is sent in the usual raster scan order, and used to overpaint the existing image. First, level 0 (1x1) is painted as a single block, covering the entire screen. Then level 1 (MxN) is sent and displayed, again filling the entire screen. Successive levels, requiring ever increasing amounts of time to transmit and display, serve to continually refine the details of the image on the screen (see Figures 2 and 3).

This method can be used to display any pyramid data structure, regardless of the choice of reduction window size and reduction rule. However, since each level is sent in its entirety, all of the effort devoted to sending levels 0-(L-1) is "wasted" when level L completely overwrites it. When the reduction window is 2x2, this means a 33.3% increase in transmission time for the full resolution picture. Also, there must be a small amount of local (to the display) computation and state, which interprets the sequence of pixel values and keeps track of such things as the current level, the position within the current raster scan, and the size of the rectangles to be painted. A small amount of preliminary information may need to be transmitted in order to initialize this local computation. This transmission overhead is negligible, however.

The progression of images produced by the naive receiver is described by:

$$(RR,f) = \mathop{\%}_{k=0}^{L} \quad \mathop{\%}_{\substack{0 \le i < 2^k \\ 0 \le j < 2^k}} B((k,i,j),C_{k,i,j})$$

where

$$C_{k,i,j}$$

is the color (value) of the pixel (k,i,j).

The order of terms for the second repainting operator does not affect the final result in this case. However, the progression of images is affected in that the order of repainting the blocks of a single level depends on the way this operator's indices are interpreted. We can simplify this expression if we assume that k will be the slowest index to increase:

$$(RR,f) = \mathop{\%}_{\substack{0 \le k \le L \\ 0 \le i < 2^k \\ 0 \le j < 2^k}} B((k,i,j),C_{k,i,j})$$

NAIVE SENDER

```
begin "send image"
 for level := 0 step 1 until L
  do begin "send level"
      for y := 0 step 1 until (N**level)-1
       do begin "send scan line"
           for x := 0 step 1 until (M**level)-1
            do Send(Pyramid[level,x,y])
           end  "send scan line"
      end "send level"
 end "send image"
```

NAIVE RECEIVER

```
begin "receive image"
 for level := 0 step 1 until L
  do begin "receive level"
      for y := 0 step 1 until (N**level)-1
       do begin "receive scan line"
           for x := 0 step 1 until (M**level)-1
            do begin "receive pixel"
                Receive(pixel);
                x1 :=   x   * ScreenMaxX / (M**level);
                x2 := (x+1) * ScreenMaxX / (M**level)-1;
                y1 :=   y   * ScreenMaxY / (N**level);
                y2 := (y+1) * ScreenMaxY / (N**level)-1;
                SetColor(pixel);
                PaintRectangle(x1,y1,x2,y2);
                end  "receive pixel"
            end  "receive scan line"
      end  "receive level"
 end  "receive image
```

EXPLICIT REPAINTING

The previous method used a fixed order of pixel transmission, and used knowledge about this order to avoid sending any positioning information. In this method, we take the view that the spatial coherence of the image is such that there are large, homogeneous areas which, once painted correctly in a low resolution image, need not be overpainted. This is often the case in binary images, and becomes less probable as the grey scale or color resolution is increased.

The transmitted information consists of a sequence of quadruples (k,i,j,v), where:

k = Level number,

i,j = Coordinates of a pixel at that Level,

v = The Color (or Value) of that pixel.

Once again, we transmit the pyramid from the top down, except that we only send a quadruple for a pixel if its Value is different than the Value of its Father (in the previous Level). The displayed images will be the same as those produced by the Naive method, but the display speed will depend strongly upon the "pyramid complexity" [Tanimoto, 1977] of the image.

This method is most likely to be useful when the range of pixel values is small, and when the pyramid is grown by a reduction rule such as Mode (Value of Father = most common Value of Sons). In these cases, we are guaranteed to have correctly painted at least 1/(M*N) of the pixels at a given level, since at least one Son in every reduction window has the same value as the Father of that window.

This method is unsuitable for the display of pyramids grown by reduction rules (e.g., Mean) which do not guarantee exact matches between Father and Son pixel values. In these cases, the overhead involved in specifying the level and coordinates of each pixel will outweigh the savings made by not sending every pixel.

A special case arises when the images are binary (0/1). In that case we are guaranteed that at least 1/2 of the pixels at a given level are already correct. In addition, we can dispense with the fourth element of every quadruple. We adopt the convention that the screen is originally blank (say, all 0's). We may refer to this "Zero image" as level -1. Then, the Value of a given quadruple is uniquely determined. Hence, it need not be sent. Instead, the meaning of the triple (k,i,j) becomes "complement the block corresponding to position (i,j) at level k" [Tanimoto, 1977].

Of course, if we do not restrict ourselves to pyramid data structures, there is a large class of successive refinement display methods based on the use of smaller and smaller rectangular (or other shaped) blocks. The tradeoffs are much the same as those addressed by divide-and-conquer hidden surface algorithms [Warnock, 1969]. Note that the (k,i,j) triple is smaller than the (X1,Y1,X2,Y2) quadruple needed to specify an arbitrarily placed rectangular block, but that arbitrary placement allows faster localization of edges which do not lie on the pyramid's reduction window boundaries. Allowing arbitrary placement of blocks also raises the question of efficient methods of determining an optimal painting sequence. Such considerations are beyond the scope of this work.

We describe the progression of images produced by the explicit repainting receiver as follows:

$$(RR,f) = \mathop{\underline{\%}}_{q=1}^{n_q} F((k_q,i_q,j_q),c_q)$$

Here the sequence of quadruples sent by the explicit repainting sender is represented by:

$$(k_1,i_1,j_1,c_1),\ (k_2,i_2,j_2,c_2),\ldots,\ (k_{n_q},i_{n_q},j_{n_q},c_{n_q}) \ .$$

In the aforementioned special case where binary (black/white) images are concerned (and the value of a given quadruple need not be sent, since it is implicitly the opposite of its father's value), we describe the explicit repainting receiver's actions as

$$(RR,f) = B(0,0,0),\ \text{black})\ \underline{\text{¢}} \left(\mathop{\underline{\text{¢}}}_{q=1}^{n_q} B((k_q,i_q,j_q),\ \text{white}) \right)$$

Thus, successive refining in this case is equivalent to successive complementation of the color of subblocks.

EXPLICIT REPAINTING SENDER

```
begin "send image"
 send(0,0,0,Pyramid[0,0,0]);
 for level := 1 step 1 until L
  do begin "send level"
      for y := 0 step 1 until (N**level)-1
       do begin "send scan line"
           for x := 0 step 1 until (M**level)-1
            do begin "send pixel"
                father := Pyramid[level-1,x/M,y/N];
                son    := Pyramid[level,x,y];
                if son NEQ father
                 then Send(level,x,y,son)
                end  "send pixel"
           end  "send scan line"
      end  "send level"
 end  "send image"
```

EXPLICIT REPAINTING RECEIVER

```
begin "receive image"
 while TRUE
  do begin "another pixel"
      Receive(level,x,y,pixel);
      x1 :=    x    * ScreenMaxX / (M**level);
      x2 := (x+1)  * ScreenMaxX / (M**level)-1;
      y1 :=    y    * ScreenMaxY / (N**level);
      y2 := (y+1)  * ScreenMaxY / (N**level)-1;
      SetColor(pixel);
      PaintRectangle(x1,y1,x2,y2)
      end  "another pixel"
 end  "receive image"
```

OMIT REDUNDANT PIXELS (SUM)

The Naive method made use of a specific ordering of the pixels in the pyramid, and the Explicit Repainting method used knowledge (on the Sender's part) about previously sent pixels in order to avoid sending "redundant" information. In this method, we rely upon knowledge on the Receiver's part about the values of previously sent pixels and the reduction rule used in growing the pyramid.

Assume we are working with scalar pixel values (color is handled by assuming we have three scalar-valued images). Suppose that the reduction rule was Sum (Value of Father = Sum of Values of Sons). We first note that each level of the pyramid requires a different number of bits to represent each pixel. When the reduction window is 2x2, level k-1 requires 2 more bits per pixel than level k. Next, we see that if we know the values of the Father and all but one of the Sons, then we can derive the value of the last Son. For example, with a 2x2 reduction window,

Son[1,1] = Father - (Son[0,0]+Son[0,1]+Son[1,0])

Now, if the values of previously sent pixels are readily available to the Receiver (i.e., the current image is stored in local memory which can be read by the Receiving process), then we can transmit the image as in the Naive method, except that we omit the last pixel in each reduction window. The Receiving process must appropriately scale all values for display purposes, and compute the values of the "missing" pixels.

Note that the Receiver may take advantage of the increased grey-scale resolution in the lower spatial resolution levels. For example, an image which is binary at level L can be displayed as a grey-scale image at earlier levels. This can be done either by using a grey-scale display device or by using half-toning techniques to shade the rectangular blocks on a binary display. This use of extra grey-scale resolution may significantly improve the early approximations to the final image.

Since we omit one pixel in each reduction window, the total number of pixels transmitted is X*Y, the number of pixels in level L. However, since pixels at different levels require more bits to specify, there is some transmission overhead involved. The absolute overhead is independent of the number of bits per pixel in level L. For a 2x2 reduction window and 12 bits of information for each (level L) pixel, the transmission overhead is 8.3%.

The progression of images produced by this receiver is identical to that produced by the naive receiver.

OMIT REDUNDANT PIXELS (SUM) SENDER

```
begin "send image"
 for level := 0 step 1 until L
  do begin "send level"
      for y := 0 step 1 until (N**level)-1
       do begin "send scan line"
            for x := 0 step 1 until (M**level)-1
             do begin "send pixel"
                 if ((y MOD N) NEQ N-1)
                     OR ((x MOD M) NEQ M-1)
                     OR (level = 0)
                  then Send(Pyramid[level,x,y])
                 end  "send pixel"
            end  "send scan line"
      end  "send level"
 end  "send image"
```

OMIT REDUNDANT PIXELS (SUM) RECEIVER

```
begin "receive image"
 for level := 0 step 1 until L
  do begin "receive level"
      for y := 0 step 1 until (N**level)-1
       do begin "receive scan line"
            for x := 0 step 1 until (M**level)-1
             do begin "receive pixel"
                 if ((y MOD N) NEQ N-1)
                     OR ((x MOD M) NEQ M-1)
                     OR (level = 0)
                  then Receive(pixel)
                  else pixel := Father(x,y)
                               - SumPreviousSons(x,y);
                 SaveValue(level,x,y,pixel);
                 SetColor(pixel/((M*N)**level));
                 x1 :=    x    * ScreenMaxX / (M**level);
                 x2 := (x+1)  * ScreenMaxX / (M**level)-1;
                 y1 :=    y    * ScreenMaxY / (N**level);
                 y2 := (y+1)  * ScreenMaxY / (N**level)- 1;
                 PaintRectangle(x1,y1,x2,y2)
                 end  "receive pixel"
            end  "receive scan line"
      end  "receive level"
 end  "receive image"
```

OMIT REDUNDANT PIXELS (SELECTION)

The previous method can be generalized to deal with any reduction rule which allows the derivation of the Value of a single Son pixel, given the Values of the Father and the remaining Sons. In particular, if the reduction rule is Selection (Value of Father = Value of Son[x',y']), then not only can we avoid sending Son[x',y'], but we do not even have to derive its value! When the other Sons are transmitted and painted on the screen, Son[x',y'] is already correctly painted on the screen. The area corresponding to Son[x',y'] was painted when the Father was painted, and does not need to be repainted. The important point is that both the Sender and the Receiver can know this. As above, we must transmit X*Y pixels. However, due to our choice of reduction rule, all pixels require the same number of bits. This means that there is absolutely no transmission overhead, compared with a row-by-row painting of level L. The advantages of early presentation to the user of a complete, albeit low resolution, image are obtained at the price of a small amount of computational overhead. Also, it is not necessary to store the image in fast memory accessible to the Receiver, since no operations other than display are required.

The values transmitted correspond exactly to the values of the pixels at level L. The order in which they are sent is the only difference between this method and the traditional row-by-row raster scan. Just as the Receiver must understand the ordering of the usual raster scan, the Receiver for this method must understand, and properly interpret, this ordering. If the time to write a large rectanqular area on the screen (or in a frame buffer) is "free" compared with the transmission time, then this method is "free".

OMIT REDUNDANT PIXELS (SELECTION) SENDER

```
begin "send image"
 for level := 0 step 1 until L
  do begin "send level"
      for y := 0 step 1 until (N**level)-1
       do begin "send scan line"
           for x := 0 step 1 until (M**level)-1
            do begin "send pixel"
                if ((y MOD N) NEQ 0) OR ((x MOD N) NEQ 0)
                    OR (level = 0)
                 then Send(Pyramid[level,x,y])
                end  "send pixel"
           end  "send scan line"
      end  "send level"
 end  "send image
```

OMIT REDUNDANT PIXELS (SELECTION) RECEIVER

```
begin "receive image"
 for level := 0 step 1 until L
  do begin "receive level"
      for y := 0 step 1 until (N**level)-1
       do begin "receive scan line"
           for x := 0 step 1 until (M**level)-1
            do begin "receive pixel"
                if ((y MOD N) NEQ 0) OR ((x MOD N) NEQ 0))
                    OR (level = 0)
                 then begin "overpaint with son"
                       Receive(pixel);
                       SetColor(pixel);
                       x1 :=    x    * ScreenMaxX / (M**level);
                       x2 := (x+1)  * ScreenMaxX / (M**level)-1;
                       y1 :=    y    * ScreenMaxY / (N**level);
                       y2 := (y+1)  * ScreenMaxY / (N**level)-1;
                       PaintRectangle(x1,y1,x2,y2)
                       end "overpaint with son"
                 end  "receive pixel"
           end  "receive scan line"
      end  "receive level"
 end  "receive image"
```

INTERACTIVE DETAILING

All of the above methods can be modified to allow the observer to direct the successive refinement process. Once the entire image has been painted to some minimum resolution, the user may interrupt the transmission of the image and indicate an area to be refined further. The refinement process is then limited to that area of the image. This will prevent the transmission of information about areas of the image which are uninteresting to the user, and allow much faster refinement of the important details.

The Explicit Repainting scheme is the easiest one to modify, since the position and extent of each rectangular block is completely specified. The other methods rely upon a fixed, known order of pixel values, and must be extended to deal with interruptions. After each user-specified windowing operation, a small amount of bookkeeping information must be transmitted, to re-initialize the Receiver.

TRANSFORM METHODS

All of the methods discussed above yield a "series" representation of the image, and have the "prefix property". That is, truncating the series at any point gives an approximation to the original image. There are, of course, other representations with this property. Two which have been used extensively in image processing are the Fourier and Hadamard transforms [Andrews, 1970]. The primary difficulty with such methods is the amount of computation required to turn the representation into a visible image. If this is to be done only once, after complete transmission of the (truncated) transform, then this might not be a serious objection. However, it is not immediately clear how to extend these methods to interactive detailing in the spatial domain.

The methods we have described have the additional property that they are well matched to the display capabilities of available raster graphics equipment. For example, painting a rectangular block is essentially free on many display devices. Also, our methods can easily be implemented requiring neither muitiplication nor division operations. Since the display equipment provides the transform inversion, this means that rapid, repeated, incremental conversion of the series representation into a viewable image is feasible.

CONCLUSION

The widespread use of high resolution raster graphics displays will require effective use of low bandwidth communication lines. We have presented several methods of transmitting raster images which provide early recognition of gross features and which are well matched to available display devices. The use of these methods is by no means restricted to display applications. They are suitable for any situation in which the Receiver can make use of a low-resolution image, especially when the required resolution is not known a priori.

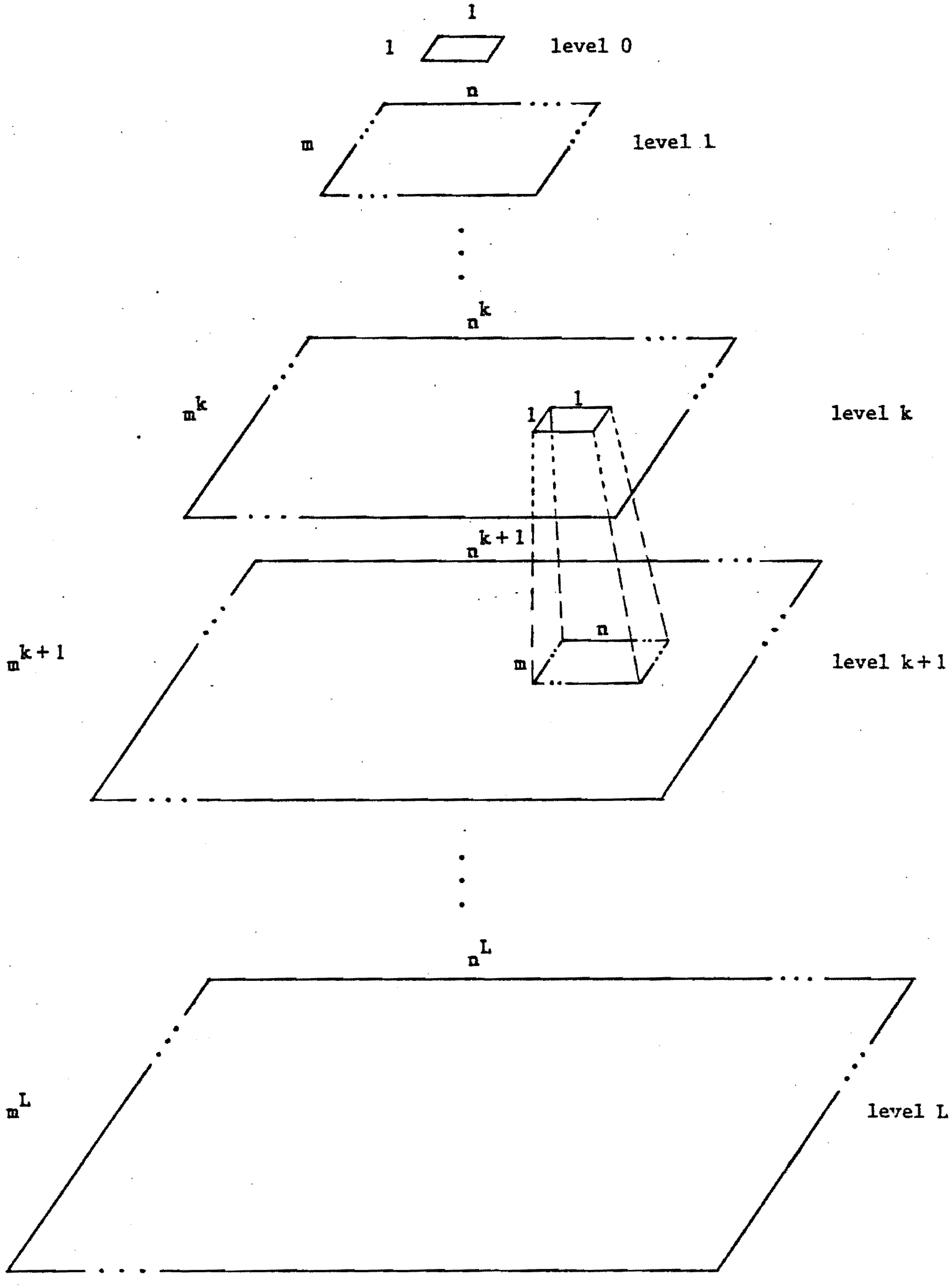
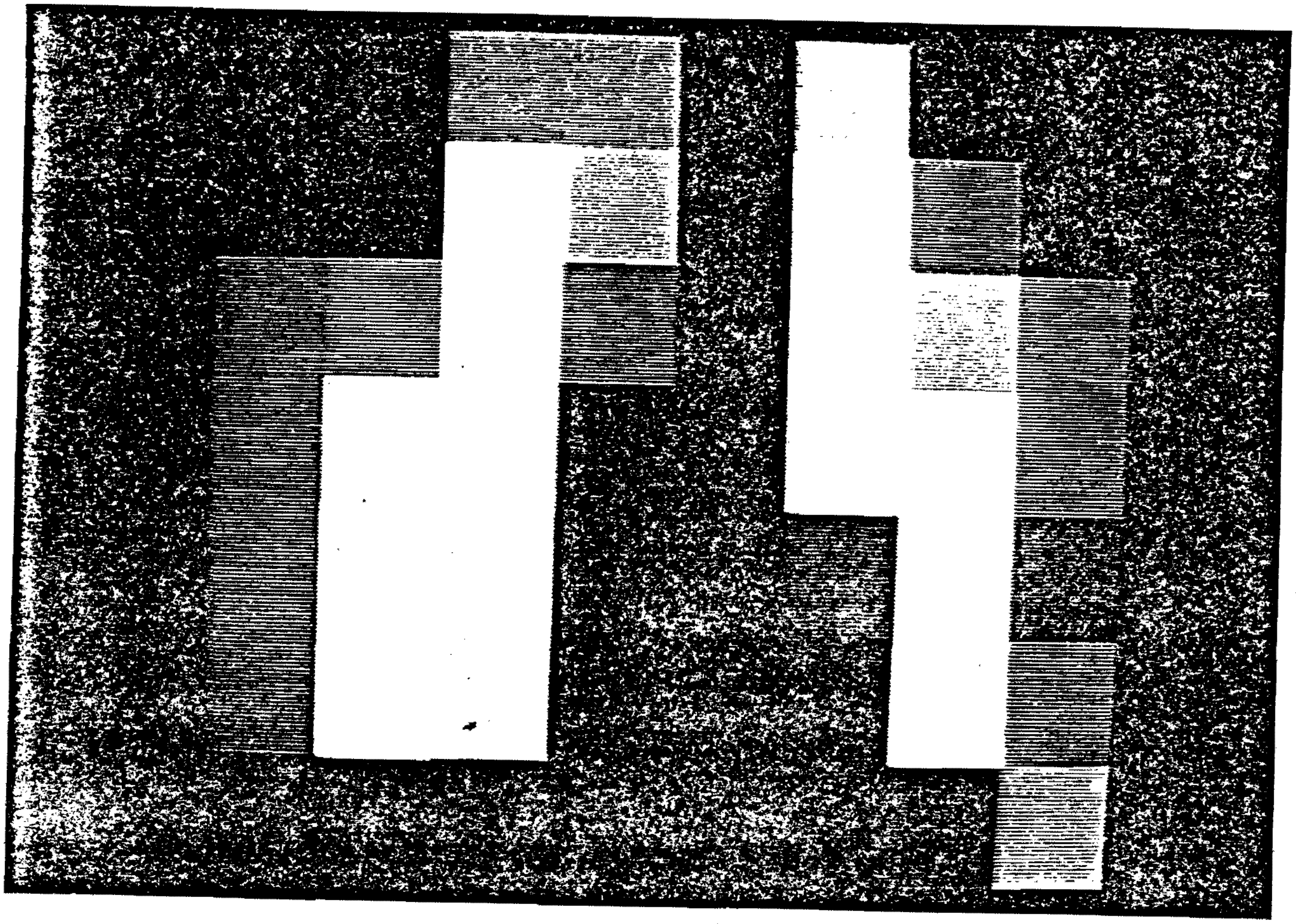
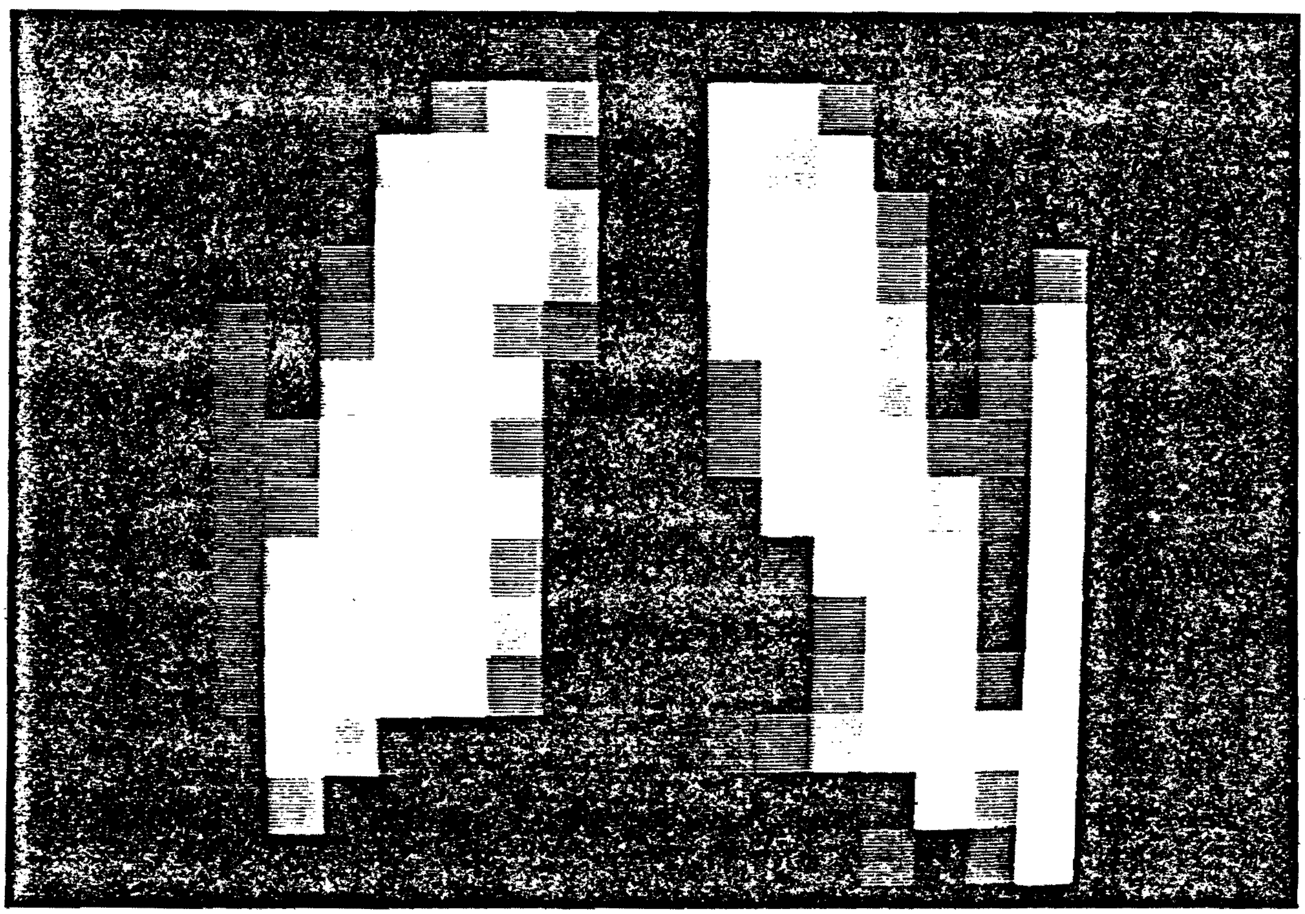
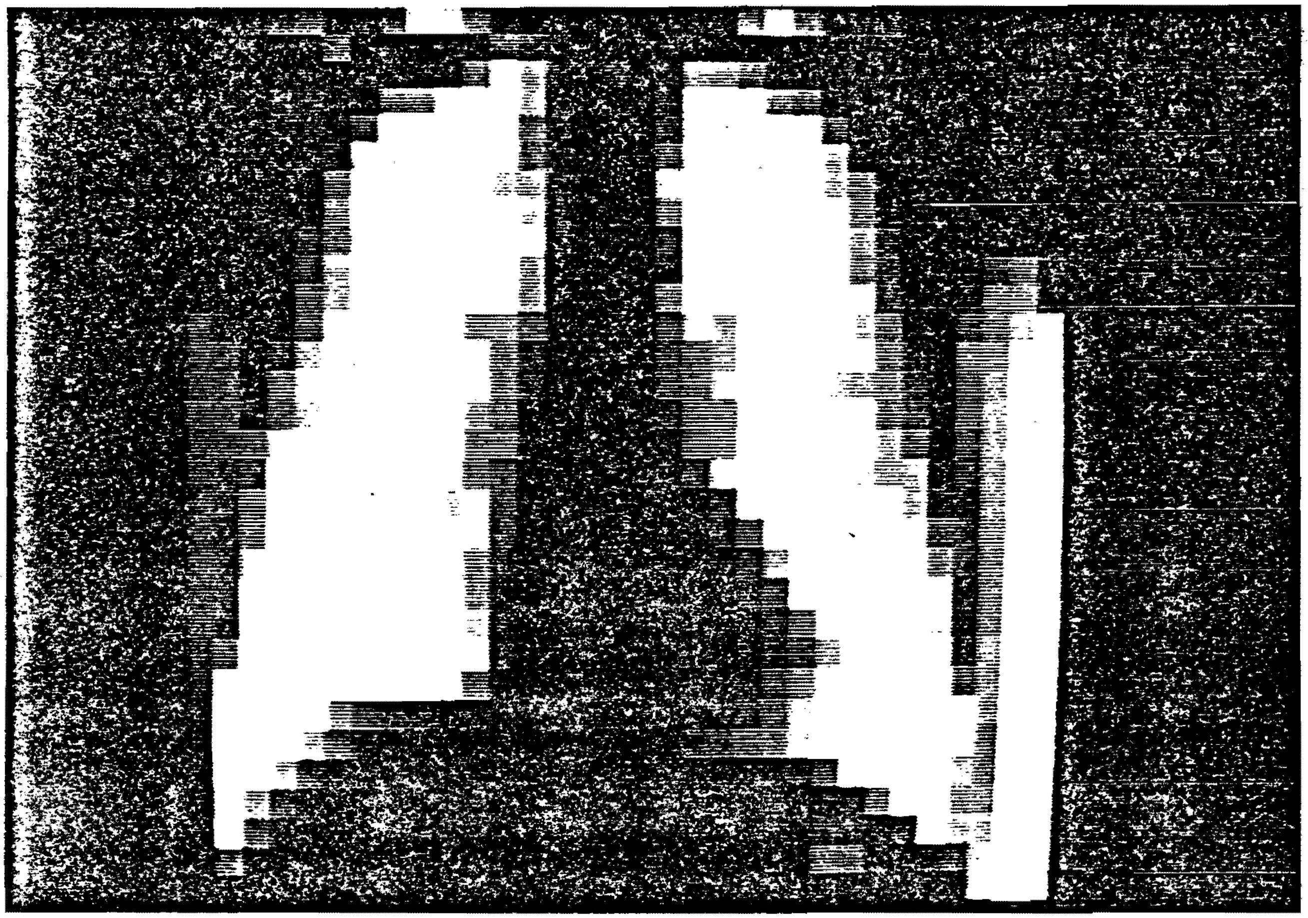
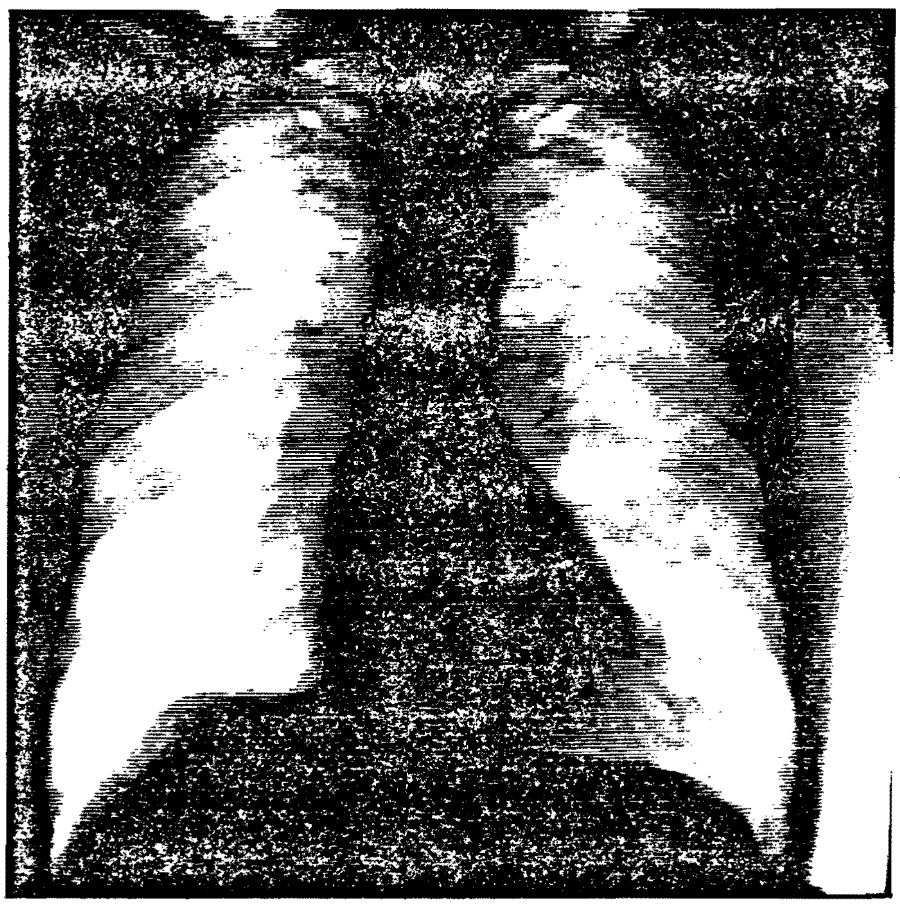


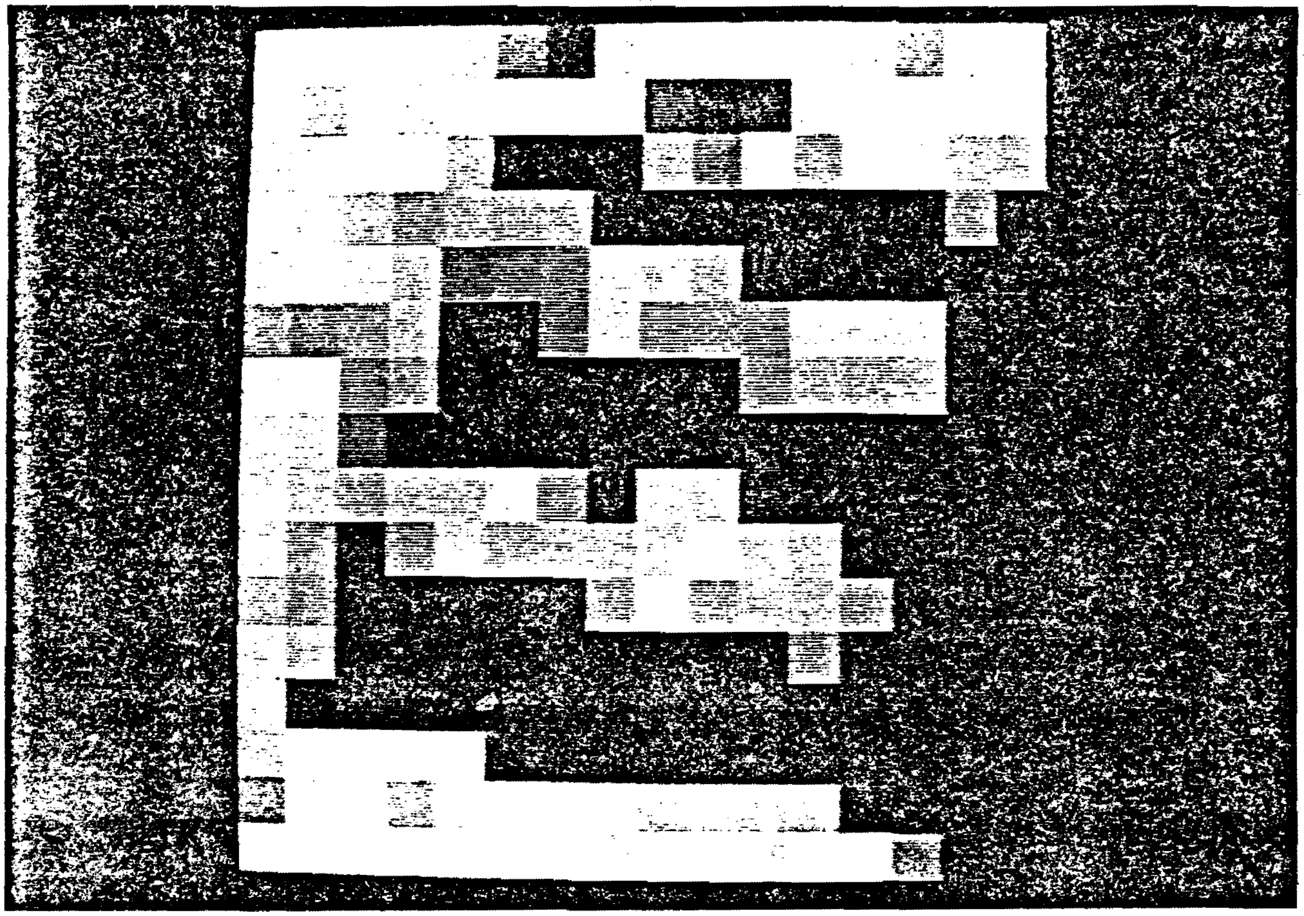
Figure 1. Pyramid data structure with $m \times n$ reduction window. Each level (e.g. level k) is an $m^k \times n^k$ pixel array where $0 \le k \le L$.

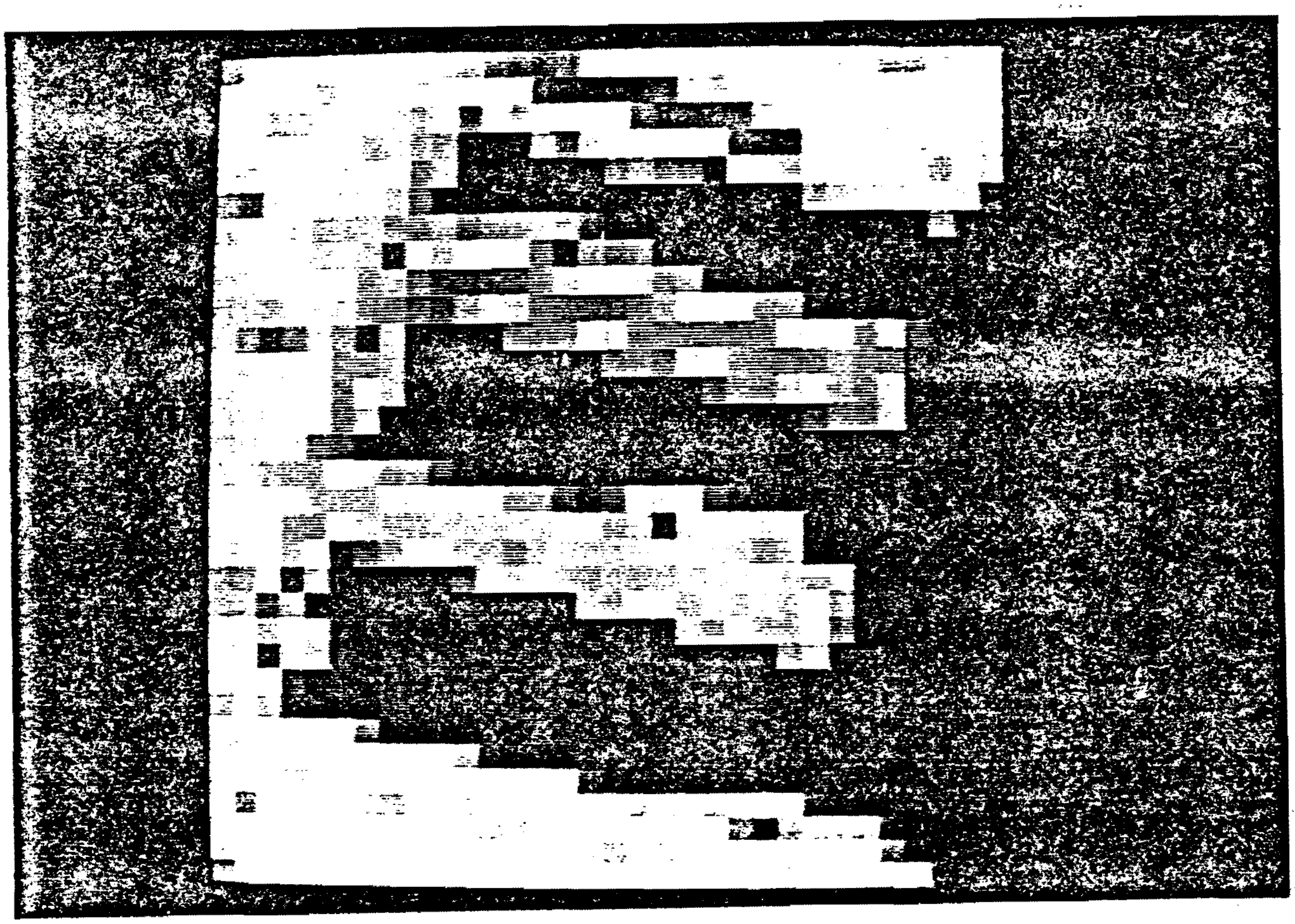


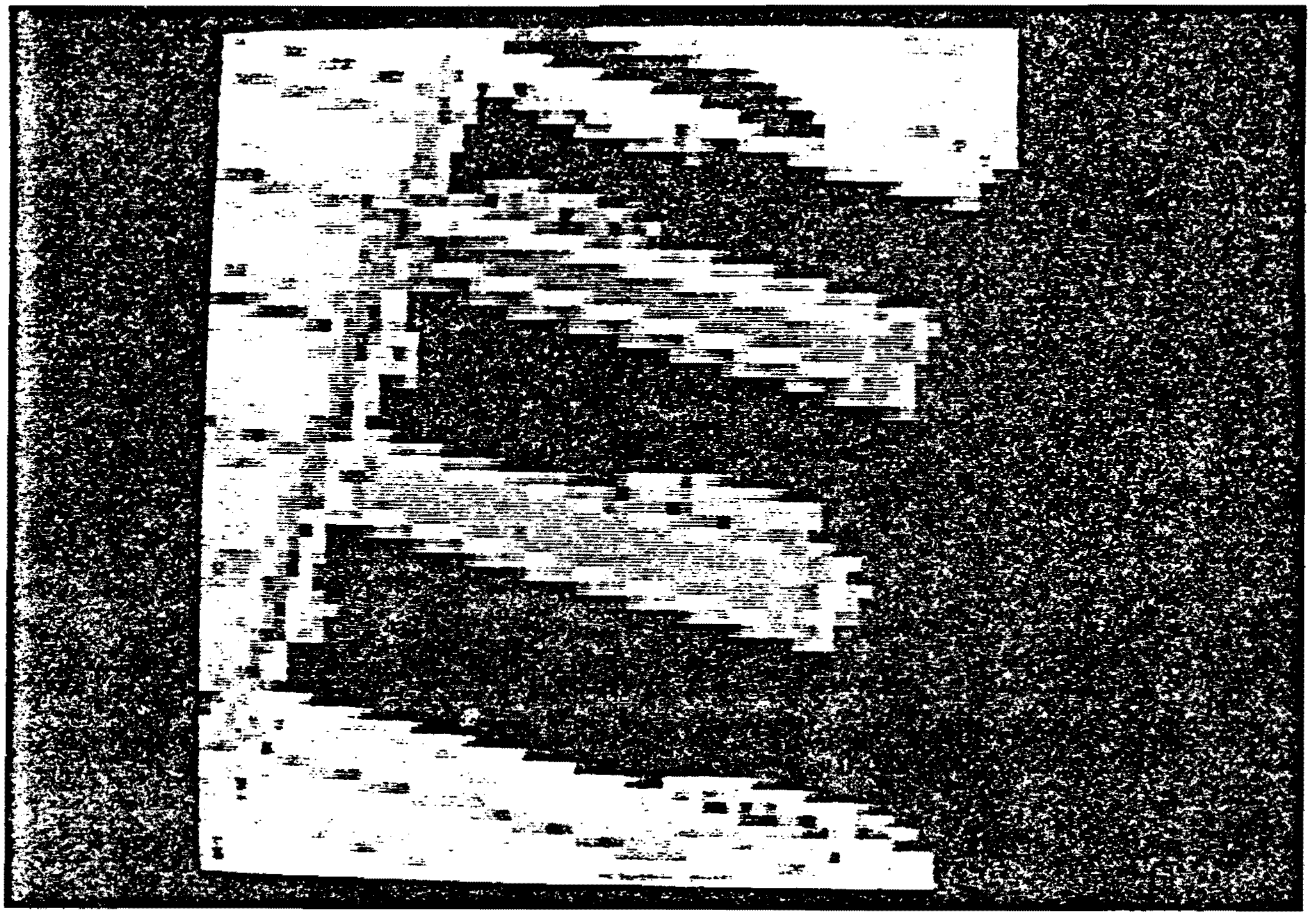


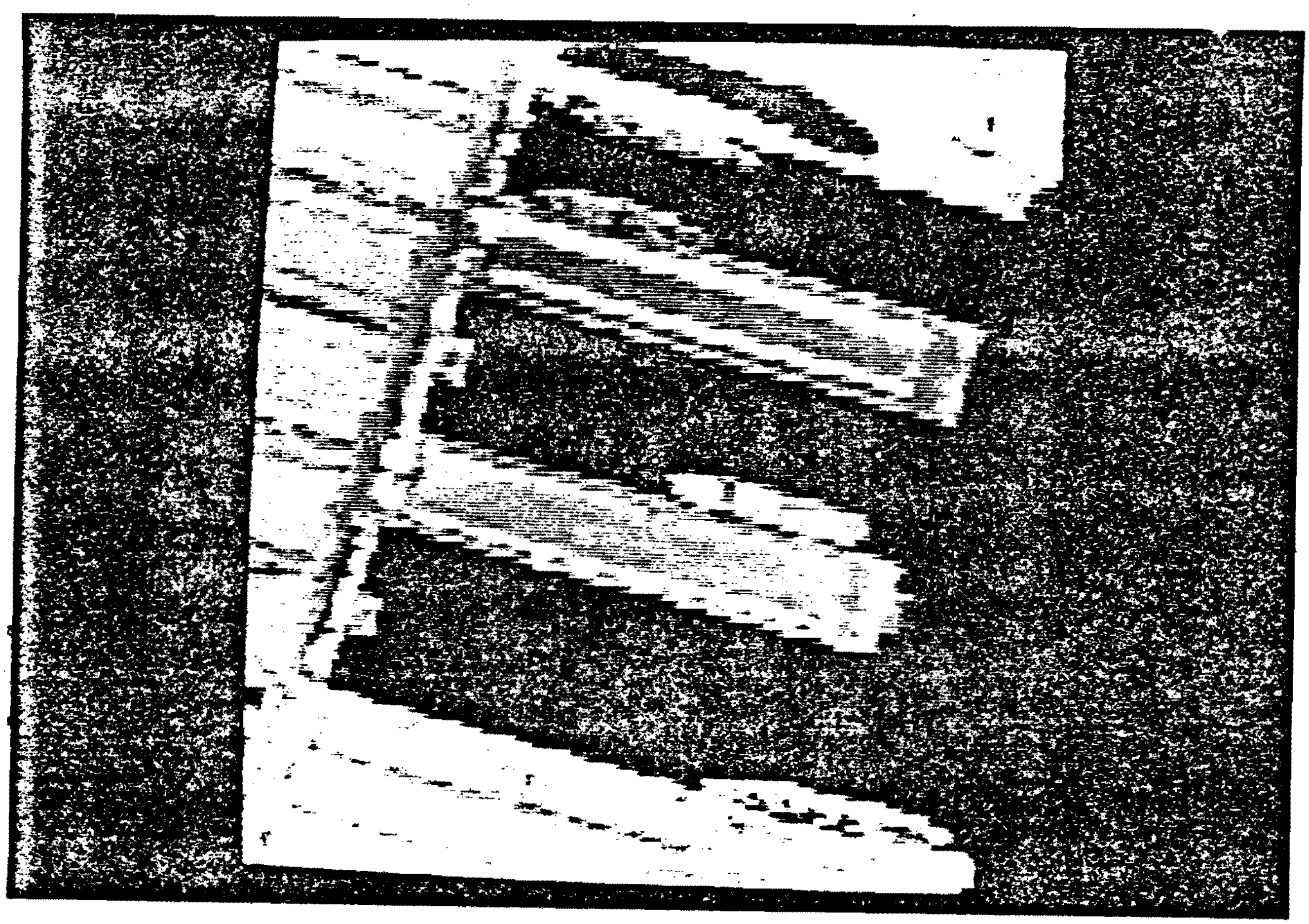


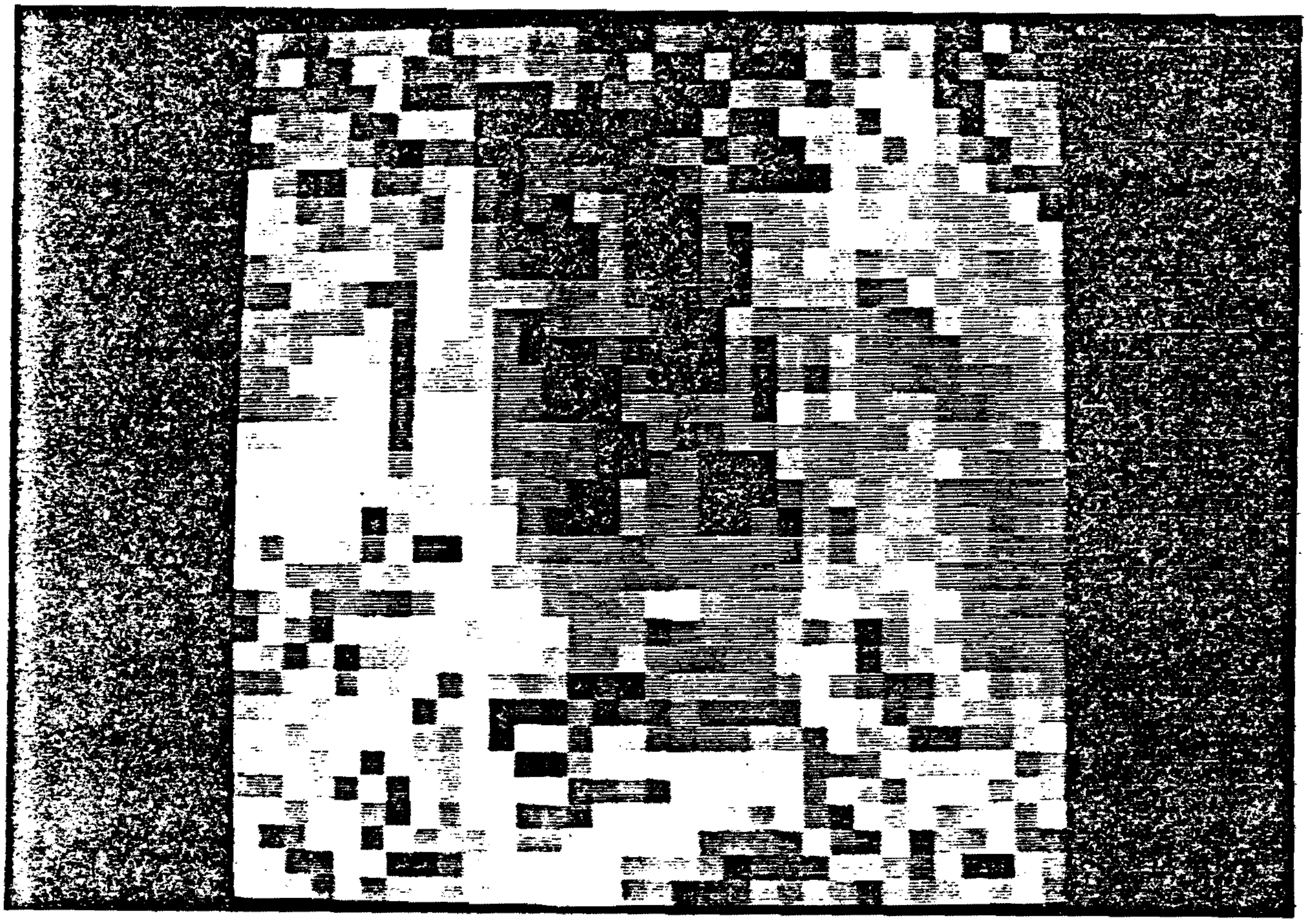


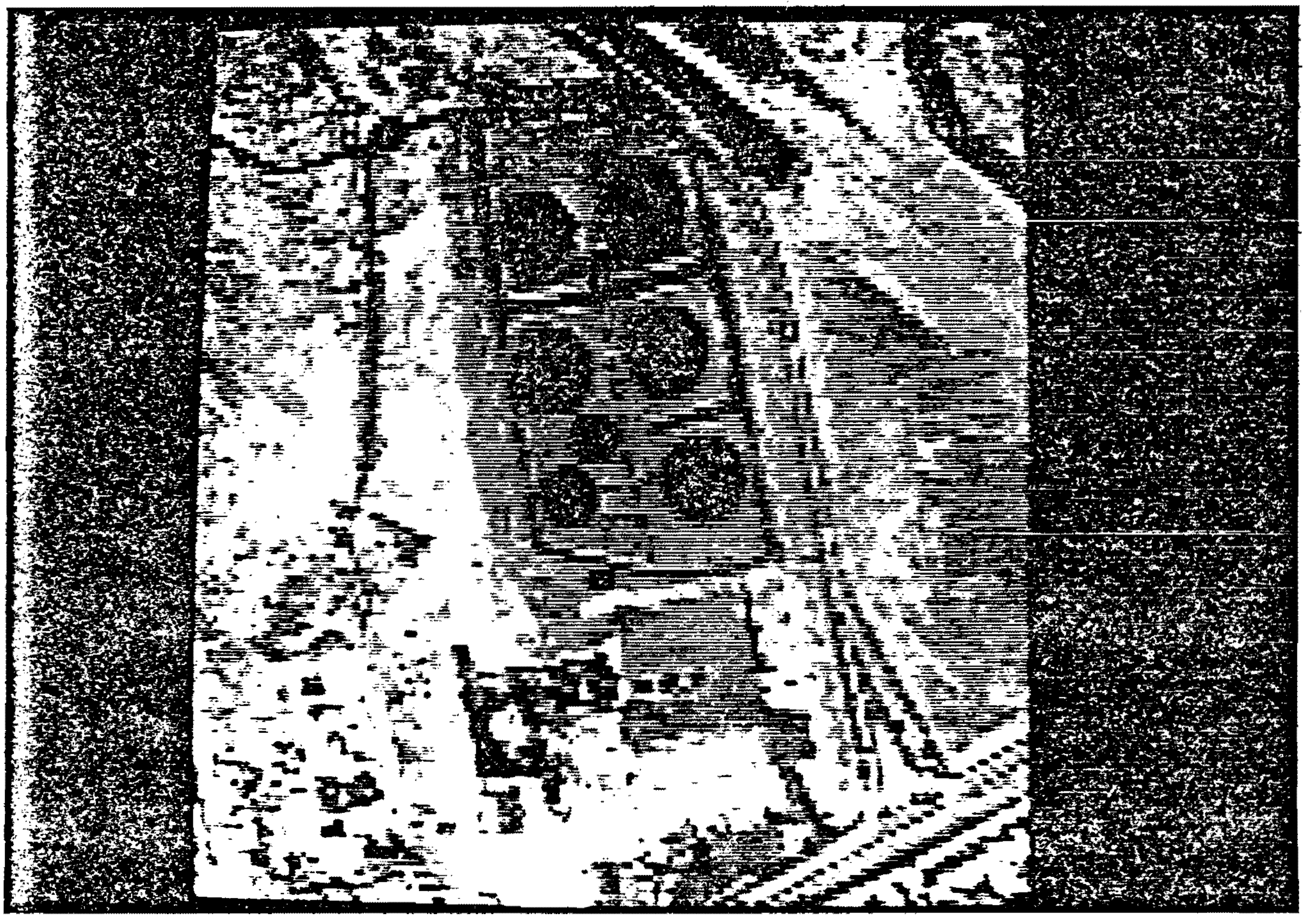


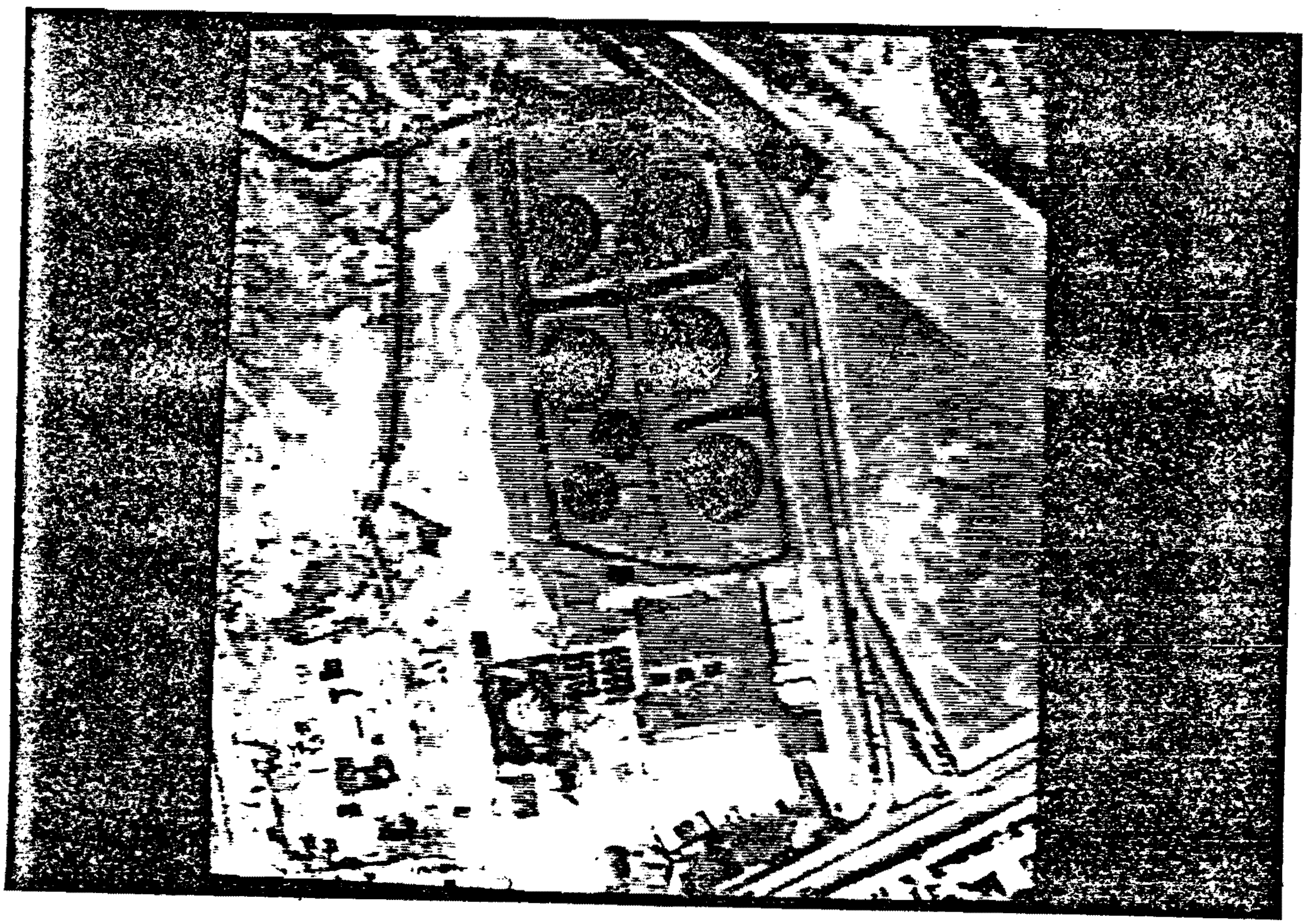












REFERENCES

Andrews, H.C. _Computer Techniques in Image Processing_. Academic Press, 1970.

Klinger, A. and C.R. Dyer. "Experiments on picture representations using regular decomposition." _Computer Graphics and Image Processing_ _5_, 1 (1976), 68-105.

Tanimoto, S.L. "A pyramid model for binary picture complexity." _Proceedings IEEE Computer Society Conference on Pattern Recognition and Image Processing_, Troy, NY (June 1977), 25-28.

Tanimoto, S.L. and T. Pavlidis. "A hierarchical data structure for picture processing." _Computer Graphics and Image Processing_ _4_, 2 (1975), 104-119.

Warnock, J.E. "A hidden-surface algorithm for computer-generated half-tone pictures." TR 4-15, Computer Science Department, University of Utah (1969).